Statistical Techniques in Robotics (16-831, S21) Lecture #13 (Wednesday, March 17)

MAB-Thompsons Sampling

Lecturer: Kris Kitani *Scribes: Yuda Song, Jinkun Cao*

1 Review

In the last lecture, we covered two multi-armed bandits algorithm: Explore-Exploit and UCB. Both algorithms can achieve sublinear regret by using concentration inequalities during the analysis. Here we give a brief review of the two algorithms.

1.1 Algorithms

The pseudocodes for Explore-Exploit and UCB are shown in Alg. 1 and Alg. 2 respectively.

```
Algorithm 1 Explore-Exploit
Require: $M$
 1: for $k = 1, \cdots, K$ do
 2:   for $m = 1, \cdots, M$ do
 3:     $a = k$
 4:     Receive($r$)
 5:     $\widehat{\mu}_k = \widehat{\mu}_k + \frac{r}{M}$
 6:   end for
 7: end for
 8: for $t = KM, \cdots, T$ do
 9:   $a^{(t)} = \arg\max_k \widehat{\mu}'_k$
10: end for
```

```
Algorithm 2 UCB
Require: $\delta, T$
 1: for $t = 1, \cdots, T$ do
 2:   if $t \leq k$ then
 3:     $k = t$
 4:   else
 5:     $k = \arg\max_{k'} \left( \widehat{\mu}_{k'} + \sqrt{\frac{\log(2T/\delta)}{2T_{k'}^{(t-1)}}} \right)$
 6:   end if
 7:   Receive($r^{(t)}$)
 8:   $T_k^t = T_k^{(t-1)} + 1$
 9:   $T_{k'}^t = T_{k'}^{(t-1)} \; \forall k' \in [K] \setminus \{k\}$
10:   $\widehat{\mu}_k = \frac{1}{T_k^{(t)}} \left( T_k^{(t-1)} \widehat{\mu}_k + r_k^{(t)} \right)$
11: end for
```

1.2 Regret Analysis

Now we review the regret bound of the two algorithms.

Theorem 1 (Regret Bound for Explore-Exploit)**.** *The regret bound of Explore-Exploit is:*

$$R = \widetilde{\mathcal{O}}(K^{1/3}T^{2/3}).$$

Theorem 2 (Regret bound for UCB)**.** *The regret bound for UCB is:*

$$R = \mathcal{O}\left(2\sqrt{\log(2T/\delta)\sqrt{KT}}\right) = \widetilde{\mathcal{O}}(\sqrt{KT}).$$

2 Bayesian Stochastic Bandit

In this lecture, we continue our setting of stochastic bandits (no adversary). And we would mainly learn Thompson Sampling [5] method.

The overall idea is that we want to keep a parametric estimation of the reward distribution. Given a prior of the distribution of the parameter, we want to update the posterior of the parameter distribution by the likelihood of the reward we witness at each round. So let's first introduce some notations here:

- $p(\theta)$: parameter prior
- $p(r|a, \theta)$: reward distribution (likelihood)
- $h^{(T)} = (a^{(t)}, r^{(t)})_{t=1}^{T}$: history of actions and rewards
- $p(\theta|h^{(t)})$: the posterior of the parameter

But why are we keeping on estimating the posterior distribution of the parameter θ? Since we assume that each reward is drawn from a parametrized distribution, which is the likelihood function:

$$r \sim p(r|a, \theta),$$

then if true parameters of likelihood function are known, the problem becomes easy: we can simply select the arm that gives the highest expected reward:

$$a = \arg\max_k \mathbb{E}_{p(r|a_k, \theta_k^*)}[r|a_k, \theta_k^*].$$

So that is the overall idea of Thompson Sampling: **we maintain a running estimate of the prior distribution hyperparameter by observing the rewards. Then we select the best arm by sampling from the estimated posterior distribution.** The form of the posterier is

$$p(\theta|h^{(t)}) = p(\theta|a^{(1)}, r^{(1)}, \ldots, a^{(t)}, r^{(t)}).$$

As the distribution is really hard to parameterize and compute, we would first make the Markov assumption over this to continue the process, which is:

$$p(r^{(t)}|\theta, a^{(t)}, a^{(t-1)}, \ldots, a^{(1)}) = p(r^{(t)}|\theta, a^{(t)}).$$

Then using Bayes rule, we have:

$$\begin{aligned}
p(\theta|h^{(t)}) &= p(\theta|a^{(1)}, r^{(1)}, \dots, a^{(t)}, r^{(t)}) \\
&= \frac{p(r^{(1)}, \dots, r^{(t)}|a^{(1)}, \dots, a^{(t)}, \theta)p(\theta|a^{(1)}, \dots, a^{(t)})}{p(r^{(1)}, \dots, r^{(t)}|a^{(1)}, \dots, a^{(t)})} \\
&= \frac{p(r^{(1)}, \dots, r^{(t)}|a^{(1)}, \dots, a^{(t)}, \theta)p(\theta)}{p(r^{(1)}, \dots, r^{(t)}|a^{(1)}, \dots, a^{(t)})} \\
&= \frac{\prod_t p(r^{(t)}|a^{(t)}, \theta)p(\theta)}{\prod_t p(r^{(t)}|a^{(t)})} \\
&\propto \prod_t p(r^{(t)}|a^{(t)}, \theta)p(\theta).
\end{aligned}$$

We note that in the formula, the form of θ can be very diverse. It can be a scalar, a tensor or even weights of a deep neural network. All depends on how the problem is formulated and the tools used to tackle the problem. Also note that we can get a recursive form of the posterior:

$$p(\theta|h^{(t)}) = p(r^{(t)}|a^{(t)}, \theta)p(\theta|h^{(t-1)}).$$

Then to get the best estimate of the parameter, we just need to find:

$$\begin{aligned}
\widehat{\theta} &= \arg\max_{\theta_k} p(\theta_k|h^{(t)}) \\
&= \arg\max_{\theta_k} p(r^{(t)}|a^{(t)}, \theta_k)p(\theta_k|h^{(t-1)}).
\end{aligned}$$

Note that line 2 of the previous equation suggests that we can perform the estimation incrementally.

3 Conjugate Priors

But how do we parameterize the likelihood, posterior and prior for efficient updates? Let's first write out the general form of the posterior:

$$\underbrace{p(\theta|x)}_{\text{posterior}} \propto \underbrace{p(x|\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}}.$$

We can see that it will be favorable if posterior and the prior have the same type of distributions. In this case, they are called conjugate distributions.
For example, the Gaussian distribution is the Conjugate prior of the Gaussian likelihood function. Another example we will show below is that the Beta distribution is the conjugate prior of the Bernoulli distribution.

3.1 Bernoulli distribution

The Bernoulli distribution is the discrete probability distribution of a random variable which takes the value 1 with probability θ and the value 0 with probability $1 - \theta$. Denote $r = \{0, 1\}$ as the random variable with a Bernoulli distribution parametrized by θ, we have

$$p(r|\theta) = \theta^r(1 - \theta)^(1 - r).$$

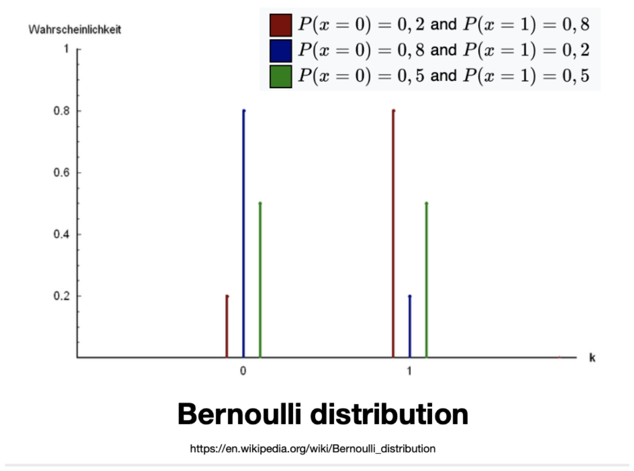


Figure 1: Bernoulli distribution

3.2 Beta distribution

Beta distribution is the probability distribution of the continuous random variable constrained in $[0, 1]$. The probability distribution is defined by two positive shape parameters, α and β. Concretely, we write:

$$p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1},$$

where the Gamma function is defined as

$$\Gamma(n) = (n-1)!.$$

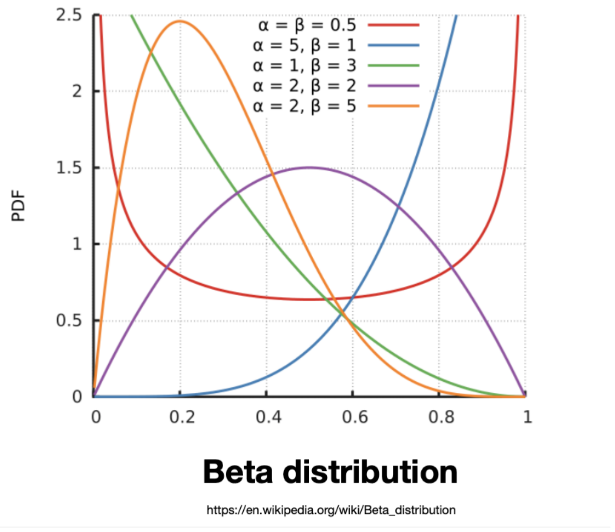


Figure 2: Beta distribution

3.3 Beta-Bernoulli Bandit

Now let's show that, in fact, the Beta distribution is the conjugate prior of the Bernoulli distribution. This will become very handy later when we want to estimate the posterior in the original problem.

Proposition 3. *The Beta distribution is the conjugate prior of the Bernoulli distribution.*

Proof. We have:

$$\begin{aligned}
\underbrace{p(\theta|r)}_{\text{posterior}} &\propto \underbrace{p(r|\theta)}_{\text{Bernoulli}} \underbrace{p(\theta)}_{\text{Beta}} \\
&\propto \theta^r(1-\theta)^{1-r}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\
&\propto \theta^{r+\alpha-1}(1-\theta)^{1-r+\beta-1} \\
&\propto \theta^{\alpha'-1}(1-\theta)^{\beta'-1},
\end{aligned}$$

where $\alpha' = r + \alpha$ and $\beta = 1 - r + \beta$. Thus we can see that the posterior is just another Beta distribution parametrized by α' and β'. □

Therefore, it is natural to form our bandit problem where we have a Beta prior on θ and the likelihood follows a Bernoulli distribution. Thus we can calculate the update rule for our estimations. Let's start with $t = 1$, we have:

$$\begin{aligned}
p(\theta|r) &\propto p(r|\theta)p(\theta) \\
&\propto \theta^r(1-\theta)^{1-r}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\
&\propto \theta^{r+\alpha-1}(1-\theta)^{1-r+\beta-1} \\
&\propto \theta^{\alpha'-1}(1-\theta)^{\beta'-1}.
\end{aligned}$$

And for time step t,

$$\begin{aligned}
p(\theta|h^{(t)}) &\propto p(r|\theta)p(\theta|h^{(t-1)}) \\
&\propto \theta^r(1-\theta)^{1-r}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\
&\propto \theta^{r+\alpha-1}(1-\theta)^{1-r+\beta-1} \\
&\propto \theta^{\alpha'-1}(1-\theta)^{\beta'-1}.
\end{aligned}$$

Thus we see that, for each case, our update rule is just

$$\alpha \leftarrow \alpha + r$$

and

$$\beta \leftarrow \beta + 1 - r.$$

4 Thompson Sampling

4.1 Algorithm

Now we are ready to introduce the algorithm. We covered three variants in the class, and we will show them in Alg. 3, Alg. 4 and Alg. 5 respectively. Note the difference between the algorithms: in batch version, we update the posterior according the likelihood of all history data, while in the incremental version, we update according the the posterior of previous timestep and the likelihood of current timestep. The Bernoulli-Beta Thompson Sampling algorithm is just an instance of incremental version of Thompson Sampling in Bernoulli-Beta Bandits settings.

Algorithm 3 Thompson Sampling (Batch Version)

```
Require: $\delta, T$
1: for $t = 1, \cdots, T$ do
2:     $\theta_k \sim p(\theta_k|h_k)\ \forall k$
3:     $a_{\widehat{k}}^{(t)} = \arg\max_k \mathbb{E}_{p(r|a_k,\theta_k)}[r]$
4:     Receive($r^{(t)}$)
5:     $h_{\widehat{k}} = h_{\widehat{k}} \bigcup (r^{(t)}, a_{\widehat{k}}^{(t)})$
6:     $p(\theta_{\widehat{k}}|h_{\widehat{k}}) \propto p(h_{\widehat{k}}|\theta_{\widehat{k}})p(\theta_{\widehat{k}})$
7: end for
```

Algorithm 4 Thompson Sampling (Incremental Version)

```
Require: $\delta, T$
1: for $t = 1, \cdots, T$ do
2:     $\theta_k \sim p(\theta_k|h_k)\ \forall k$
3:     $a_{\widehat{k}}^{(t)} = \arg\max_k \mathbb{E}_{p(r|a_k,\theta_k)}[r]$
4:     Receive($r^{(t)}$)
5:     $p(\theta_{\widehat{k}}|h_{\widehat{k}}) \propto p(r^{(t)}|a_{\widehat{k}}^{(t)}, \theta_{\widehat{k}})p(\theta_{\widehat{k}}|h_{\widehat{k}})$
6: end for
```

Algorithm 5 Thompson Sampling (Bernoulli-Beta)

```
Require: $\delta, T$
1: for $t = 1, \cdots, T$ do
2:     $\theta_k \sim p(\theta_k|h_k)\ \forall k$
3:     $a_{\widehat{k}}^{(t)} = \arg\max_k \mathbb{E}_{p(r|a_k,\theta_k)}[r]$
4:     Receive($r^{(t)}$)
5:     $\alpha_{\widehat{k}} = \alpha_{\widehat{k}} + r^{(t)}$
6:     $\beta_{\widehat{k}} = \beta_{\widehat{k}} + 1 - r^{(t)}$
7: end for
```

4.2 Performance

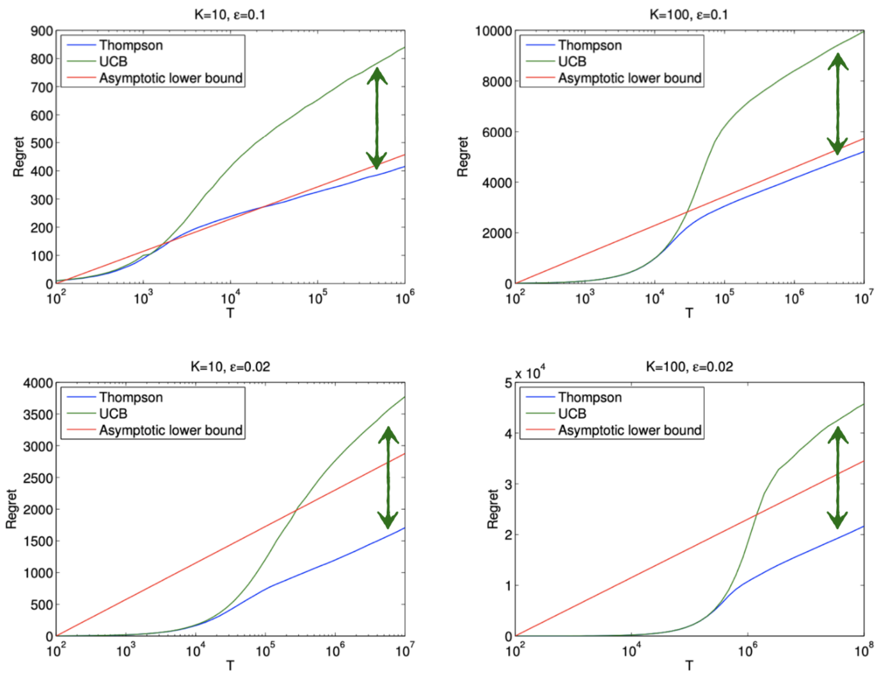
Not surprisingly, Thompson sampling is a no regret algorithm. Here we provide the regret bound:

Theorem 4 (Regret of Thompson Sampling)**.** *The regret of Thompson Sampling is*

$$R(T) = O(\sqrt{KT\log(T)}) = \widetilde{O}(\sqrt{KT}).$$

On the other hand, the literature [2] shows that Thompson sampling actually enjoys excellent empirical performance, comparing with UCB:

In this simulation, the best arm has a reward probability of 0.5 and the $K-1$ other arms have a probability of $0.5-\varepsilon$.



Thompsons Sampling has lower empirical regret especially after many time steps

Figure 1: Cumulative regret for $K \in \{10, 100\}$ and $\varepsilon \in \{0.02, 0.1\}$. The plots are averaged over 100 repetitions. The red line is the lower bound (2) shifted such that it goes through the origin.

Figure 3: Empirical performance of Thompson sampling.

5 Conclusion

In this lecture, we studied the classic Thompson Sampling algorithm under the stochastic bandit environment. In such environment, we assume that the feedback from the environment is not contextual that it basically have a fixed underlying distribution to map the action to the reward. To find a good strategy maximizing reward from the environment, it is essential to solve an expectation maximum problem with a known parametric distribution of reward while that is what we just do not know in usual situations. So the problem comes to estimating the distribution of reward by historical observations. Then, the basic philosophy of Thompson Sampling comes: by maintaining a running estimation of the prior hyperparameter of rewards with historical observation, we select the *best* arm by sampling from the estimated posterior distribution.

Thompson Sampling is proven a non-regret algorithm, but the empirical experiments [2] shows that it actually enjoys performance gains when compared with UCB.

References

[1] S. Agrawal and N. Goyal. Further optimal regret bounds for thompson sampling. In *Artificial intelligence and statistics*, pages 99–107. PMLR, 2013.

[2] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24:2249–2257, 2011.

[3] C. Riquelme, G. Tucker, and J. Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

[4] D. Russo and B. Van Roy. An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471, 2016.

[5] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[6] W. Zhang, D. Zhou, L. Li, and Q. Gu. Neural thompson sampling. *arXiv preprint arXiv:2010.00827*, 2020.

6 Appendix

There are some truly wonderful materials to help us learn about about Thompson Sampling. For example, a Stanford tutorial [1] covers more backgrounds, contents and extension following the problem and philosophy of Thomspon Sampling. Moreover, with the rise of deep learning, many researchers are looking into combining deep models with Thompson Sampling strategy.

6.1 More analysis of Thompson Sampling

For the regret bound of Thompson Sampling, there is a paper [1] provides a new path to prove the first near-optimal problem-independent bound of $O(\sqrt{NT \ln T})$ on the expected regret of this algorithm. This paper is pretty hardcore, but even more amazing to provide more advanced knowldege about Thompson Sampling.

Besides, another wonder Stanford paper [4] provides an information-theoretic analysis of Thompson sampling that applies across a broad range of online optimization problems in which a decision-maker must learn from partial feedback. In many practical cases, the analysis provided in this work might be more useful than the generic or other forms of Thompson Sampling we learned during the class.

6.2 Combining Thompson Sampling and Deep Learning

Similar to the assumption of what we made during class, a previous work [3] conducts an empirical comparison of performance in using approximate Bayesian methods under a Thompson Sampling framework in reinforcement learning. They benchmark a series of related works over different contextual bandit problems in online settings. This would be an interesting literature to read if you have interest in the direction of reinforcement learning of posterior estimation under the Thompson Sampling framework to solve online problems.

There is another work combining Thompson Sampling with deep learning [6]. The authors propose a new algorithm, called Neural Thompson Sampling, which adapts deep neural networks for both exploration and exploitation. At the core of our algorithm is a novel posterior distribution of the reward, where its mean is the neural network approximator, and its variance is built upon the neural tangent features of the corresponding neural network. They prove in the paper that, provided the underlying reward function is bounded, the proposed algorithm is guaranteed to achieve a cumulative regret of $O(T^{1/2})$, which matches the regret of other contextual bandit algorithms in terms of total round number T.

There are actually much more interesting novel application or advanced analysis of Thompson Sampling, we just show some ones we found above to provide little inspiration of leveraging Thompson Sampling, the classic strategy developed decades ago, in more modern and newly born problems with more advanced tools we are familiar with today.

[1] https://web.stanford.edu/ bvr/pubs/TS_Tutorial.pdf